We claim:

1. In a multinode computer system with distributed shared memory, a method of ensuring that a remote node receives a copy of a cache line stored on a home node, comprising:

from a remote node, requesting a shared copy of a cache line that is stored on a home node;

while waiting for the requested cache line, receiving on the remote node a request to invalidate the cache line; and

in response to the request to invalidate the cache line, requesting an exclusive copy of the cache line.

2. The method of claim 1, further including storing a shared copy of the cache line on the remote node and rolling out the cache line on the remote node prior to the request for a shared copy of the cache line, without informing the home node of the rollout.

3. The method of claim 1, wherein the remote and home nodes do not use a directory to track the locations of cache lines.

4. The method of claim 1, further including issuing a request for the cache line from a processor on the home node at approximately the same time the request for the shared copy of cache line is received from the remote node.

5. The method of claim 1, further including discarding a response to the request for the shared copy of cache line after receiving the invalidate request.

6. The method of claim 1, further including passing data between the nodes using a system interconnect that includes a dualport RAM controlled by at least one state machine.

7. The method of claim 1, further including providing a state machine in the remote node that upon requesting the cache line, remains in a first pending state;

if while in the first pending state, the cache line is received, storing the cache line into a cache on the remote node and transitioning to a dirty or fresh state;

if while in the first pending state, a request to invalidate is received, transitioning to a second pending state; and

while in the second pending state, upon receiving the cache line, discarding the cache line and issuing the request for an exclusive copy of the cache line.

8. The method of claim 1 wherein the multinode computer system includes multiple processors on each node and the multinode computer system is an unordered network.

9. The method of claim 1 wherein the multinode computer system includes at most two nodes.

10. In a multinode computer system with distributed shared memory, a method of ensuring that a remote node receives a copy of a cache line stored on a home node, comprising:

storing a copy of a cache line on the remote node, wherein the cache line is a shared copy of the cache line that is also stored on the home node;

overwriting the cache line on the remote node without informing the home node that the cache line is no longer stored on the remote node;

after overwriting the cache line, requesting a new copy of the cache line be delivered to the remote node again;

receiving, on the remote node, a request to invalidate the cache line before the requested copy of the cache line is received; and

issuing a request for an exclusive copy of the cache line so that the remote node can obtain control of the cache line.

11. The method of claim 10 further including:

receiving a response on the remote node for the requested cache line after receiving the invalidate request; and

discarding the response on the remote node for the requested cache line;

12. The method of claim 10 further including:

receiving a processor request on the home node for control of the cache line;

in response to the processor request, checking local tags to determine if another node has a shared copy of the cache line; and

upon determining that another node has a shared copy, sending the invalidate request to the remote node because the home node was not notified that the remote node has overwritten its copy of the cache line.

13. The method of claim 10 further including receiving on the remote node an exclusive copy of the cache line.

14. A multinode computer system with distributed shared memory that ensures a remote node receives a copy of data stored on a home node, comprising:

a first node having multiple processors, a local memory, and a remote cache, wherein the first node is a home node that stores a memory block of interest;

a second node having multiple processors, a local memory, and a remote cache, wherein the second node is a remote node with regard to the memory block of interest;

a system interconnect coupling the first node to the second node; and

a state machine located on the second node, the state machine moving between different states in response to requests made from the second node to the first node and responses to the requests;

the state machine having states such that when an invalidate request is received directly after a request is made for a copy of the memory block of interest, the state machine automatically sends a request for an exclusive copy of the memory block of interest.

15. The multinode computer system of claim 14 wherein the first and second nodes include snoopy cache protocol engines.

16. The multinode computer system of claim 14 wherein the first and second nodes do not use a directory or list of any kind to track the location of cache lines that are stored in cache on another node.

17. The multinode computer system of claim 14 wherein the remote node performs silent rollouts of data.

18. A multinode computer system with multiple processors on the nodes that ensures a remote node obtains control of data, comprising a state machine that monitors whether an invalidate message for data is received after issuing a request for a shared copy of the data and in response to the invalidate message issues a request for an exclusive copy of the data.

19. In a multinode computer system with distributed shared memory, a method of ensuring that a remote node receives data efficiently, comprising:

providing a first node having multiple processors, memory and a remote cache, the first node being a home node with respect to a memory block of interest and wherein the first node does not have a cache directory;

providing a second node having multiple processors, memory and a remote cache, the second node being a remote node with respect to the memory block of interest and wherein the second node does not have a cache directory;

receiving a request on the second node from a processor on the second node for the memory block of interest;

checking remote cache on the second node to determine whether the memory block of interest is stored in the remote cache;

if the memory block of interest is not stored in the remote cache, automatically assuming that the memory block of interest is stored on the first node since the second

node cannot check a cache directory to determine the location of the memory block of interest; and

sending a request to the first node for the memory block of interest.

20. The method of claim 19 further including:

receiving the memory block of interest on the second node; and

rolling out the memory block of interest on the second node without informing the first node of the rollout.